IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Matter of the Application of: John Michael Lake
Serial No.: 10/801,369 Confirmation No.: 3169
Filed: March 16, 2004
For: Determining Software Complexity
Examiner: Anil Khatri Group Art Unit: 2163
Attorney Docket No.: RSW920040039US1

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

In response to the Office Action of January 6, 2009, and in support of the Notice of Appeal file on April 6, 2009, Applicants respectfully submit this Appeal Brief.

(I) Real Party in Interest

The real party in interest for this Application is assignee INTERNATIONAL BUSINESS MACHINES CORPORATION of Armonk, NY.

(II). Related Appeals and Interferences

There are no related appeals or interferences.

(III). Status of Claims

Claims 9-19 are pending in this case. Claims 9-19 stand provisionally rejected for nonstatutory double patenting over claims 1-8 of co-pending application 11/853,017. Claims 9-19 stand rejected under 35 USC 101 as claiming non-statutory subject matter. Claims 9-19 stand rejected under 35 USC 112 as being indefinite. Claims 1-8 have been canceled. Claims 9-19 are being appealed.

(IV). Status of Amendments

An amendment filed on march 31, 2009 with a terminal disclaimer and has not been entered. No amendments were made to the claims in the amendment of March 31, 2009.

(V). Summary of claimed subject matter

(V.A) Claim 9

Claim 9 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraph 018 and paragraph 019) for determining a plurality of versions of the software component (P0, P1, P2; Steps 100, 110, 120 in Fig. 1, paragraphs

010, 013, 014, 019) and for finding lengths of compressed versions of the plurality of versions of the software (C0, C1, C2, Steps140, 150, 160 in Fig. 1, paragraphs 010, 016, 019). The apparatus also comprises means for compressing each of the versions (210 in Fig. 2, paragraphs 018, 019) to provide the compressed versions, and means for comparing the lengths of the compressed versions (220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019). A display, other output device or equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(V.B) Claim 10

Claim 10 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraphs 018, 019) for creating raw program text (P0; Step 100 in Fig. 1, paragraphs 013, 014, 019) and normalized program text (P1; step 110 in Fig. 1, paragraphs 013, 014, 019) of the software component and for finding lengths of compressed raw program text (C0; Step 140 in Fig. 1, paragraphs 016, 019) and compressed normalized program text (C1; Steps 150 in Fig. 1, paragraphs 016, 019). The apparatus also comprises means for compressing the raw program text and the normalized program text (210 in Fig. 2, paragraphs 018, 019) to provide the compressed versions (Step 1130 in Fig. 1, paragraphs 018), and means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text (220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.C) Claim 11

Claim 11 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraphs 018, 019) for creating normalized program text (P1; step 110 in Fig. 1, paragraphs 013, 014, 019) and normalized unique program text (P2; Step120 in Fig. 1, paragraphs 013, 014, 019) of the software component and for finding

lengths of compressed normalized program text (C1; Steps 150 in Fig. 1, paragraphs 016, 019) and compressed normalized unique program text (C2; Step 160 in Fig. 1, paragraphs 016, 019). The apparatus also comprises means for compressing the normalized program text and the normalized unique program text (210 in Fig. 2, paragraphs 018, 019) to provide the compressed versions (Step 130 in Fig. 1, paragraphs 018), and means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text (C1/C2; 220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019). A means for providing a software complexity metric comprising the ratio of the lengths of the compressed versions is provided by a display or other output device or equivalent (paragraphs oo0, 017, 020).

(V.D) Claim 12

Claim 12 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component (paragraph 020). The program of instruction creates a plurality of versions of the software component (P0, P1, P2; Steps 100, 110, 120 in Fig. 1, paragraphs 010, 013, 014, 019). The program of instruction compresses each of the versions, to provide compressed versions (C0, C1, C2; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 140, 150, 160 in Fig. 1, paragraphs 016, 019). The program of instruction compares the lengths of the compressed versions (Step 170 in Fig. 1, paragraphs 006, 010, 016, 017, 019). The program of instructions provides a software complexity metric comprising a comparison of the lengths of the compressed versions is provided by a display or other output device or equivalent (paragraphs oo0, 017, 020).

(V.F) Claim 13

Claim 13, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises raw program text (PO; Step 100 in Fig. 1, paragraphs 013, 014, 019).

(V.F) Claim 14

Claim 14, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises normalized program text (P1; Step 110 in Fig. 1, paragraphs 013, 014, 019).

(V.G) Claim 15

Claim 15, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises normalized unique program text (P2; Step 120 in Fig. 1, paragraphs 013, 014, 019).

(V.H) Claim 16

Claim 16, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the comparison of compressed versions comprises finding a ratio using the length of the compressed version of raw program text and the length of the compressed version of normalized program text (C0/C1; Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.I) Claim 17

Claim 17, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein wherein the comparison of compressed versions comprises finding a ratio using the length of the compressed version of normalized program text and the length of the compressed version of normalized unique program text C1/C2; Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.J) Claim 18

Claim 18 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component (paragraph 020). The program of instruction creates creating raw program text (P0; Step 100 in Fig. 1, paragraphs 010, 013, 014, 019) and normalized program text (P1; Step 110 in Fig. 1, paragraphs 010, 013, 014, 019) of the software component. The program of instruction compresses the raw program text and the normalized program text, to provide compressed versions (C0, C1; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 140, 150 in Fig. 1, paragraphs 016, 019). The program of instructions finds a ratio of the length of the compressed raw program text to the length of the compressed normalized program text (C0/C1; Step 170 in Fig. 1, paragraphs 006, 010, 016, 017, 019) and provides a software complexity metric comprising the ratio (paragraphs 006, 017, 020).

(V.K) Claim 19

Claim 19 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component (paragraph 020). The program of instruction creates normalized program text (P1; Step 110 in Fig. 1, paragraphs 010, 013, 014, 019) and normalized unique program text (P2; Step 120 in Fig. 1, paragraphs 010, 013, 014, 019) of the software component. The program of instruction compresses the normalized program text and the normalized unique program text, to provide compressed versions (C1, C2; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 150, 160 in Fig. 1, paragraphs 016, 019). The program of instructions finds a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text (C1/C2; Step 170 in Fig. 1,

paragraphs 006, 010, 016, 017, 019) and provides a software complexity metric comprising the ratio (paragraphs 006, 017, 020).

(V.L) Corresponding Structure for Means of Claim 9

The structure corresponding to means for compressing each of the versions is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for comparing the lengths of the compressed versions is a divider such as a microprocessor and equivalents (paragraph 018). A display other output device or equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(V.M) Corresponding Structure for Means of Claim 10

The structure corresponding to means for compressing the raw program text and the normalized program text is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text is a divider such as a microprocessor and equivalents (paragraph 018).

(V.N) Corresponding Structure for Means of Claim 11

The structure corresponding to means for compressing normalized program text and the normalized unique program text is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text is a divider such as a microprocessor and equivalents (paragraph 018). A display other output device or

equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(VI). Grounds of Rejection to be reviewed on appeal

Each of claims 9-19 is rejected for nonstatutory double patenting over claims 1-8 of co-pending application 11/853,017, under 35 USC 101 as claiming non-statutory subject matter, and under 35 USC 112 as being indefinite.

The questions for appeal are: Whether or not the terminal disclaimer filed on March 31, 2009 should be entered to overcome the provisional rejection for non-statutory double patenting; whether or not each of claims 9-19 claims non-statutory subject matter under 35 USC 101; and whether or not each of claims 9-19 is indefinite under 35 USC 112, second paragraph.

(VII). Argument

(VII.A) Principles of Law Relating to Non-statutory Double Patenting

The judicially created doctrine of double patenting seeks to prevent the unjustified extension of patent exclusivity beyond the term of a patent. The public policy behind this doctrine is that:

> The public should . . . be able to act on the assumption that upon the expiration of the patent it will be free to use not only the invention claimed in the patent but also modifications or variants which would have been obvious to those of ordinary skill in the art at the time the invention was made, taking into account the skill in the art and prior art other than the invention claimed in the issued patent.

In re Zickendraht, 319 F.2d 225, 232, 138 USPQ 22, 27 (CCPA 1963) (Rich, J., concurring). Double patenting results when the right to exclude granted by a first patent is unjustly extended by the grant of a later issued patent or patents. *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982). A rejection based on a nonstatutory type of double patenting can be avoided by filing a terminal disclaimer in the application or proceeding in which the rejection is made. *In re Vogel,* 422 F.2d 438, 164 USPQ 619 (CCPA 1970); *In re Knohl*, 386 F.2d 476, 155 USPQ 586 (CCPA 1967); and *In re Griswold*, 365 F.2d 834, 150 USPQ 804 (CCPA 1966).

(VII.B) Principles of Law Relating to Patentable Subject Matter

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title. 35 USC 101. As the Supreme Court has recognized, Congress chose the expansive language of 35 U.S.C. 101 so as to include "anything under the sun that is made by man" as statutory subject matter. *Diamond v. Chakrabarty*, 447 U.S. 303, 308-09, 206 USPQ 193, 197 (1980). The phrase "anything under the sun that is made by man" is limited by the text of 35 U.S.C. 101, meaning that one may only patent something that is a machine, manufacture, composition of matter or a process. *Alappat*, 33 F.3d at 1542, 31 USPQ2d at 1556.

When functional descriptive material is recorded on some computer-readable medium, it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994)(discussing patentable weight of data structure limitations in the context of a statutory claim to a data structure stored on a computer readable medium that increases computer efficiency) and >*In re*< *Warmerdam*, 33 F.3d *>1354,< 1360-61, 31 USPQ2d *>1754,< 1759 (claim to computer having a specific data structure stored in memory held statutory product-by-process claim) with *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* held nonstatutory). Zccordingly the MPEP instructs that when a computer program is recited in conjunction with a physical structure, such as a computer memory, USPTO personnel should treat the claim as a product claim. MPEP 2601.01.

Whether or not a process claim is directed to patentable subject matter is dertermined by the machine-or-transformation test. In re Bilski _ F. 3d _ (Fed. Cir 2008)(en banc). The machine-or-transformation test is a two-branched inquiry; an applicant may show that a process claim satisfies § 101 either by showing that his claim is tied to a particular machine, or by showing that his claim transforms an article.

(VII.C) Principles of Law Relating to Definiteness

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention. 35 USC 112,

second paragraph. The primary purpose of the requirement in the second paragraph of 35 USC 112 for definiteness of claim language is to ensure that the scope of the claims is clear so the public is informed of the boundaries of what constitutes infringement of the patent. (See MPEP § 2173).

"The requirement to 'distinctly' claim means that the claim must have a meaning discernible to one of ordinary skill in the art when construed according to correct principles....Only when a claim remains insolubly ambiguous without a discernible meaning after all reasonable attempts at construction must a court declare it indefinite.").*Metabolite Labs., Inc. v. Lab. Corp. of Am. Holdings*, 370 F.3d 1354, 1366, 71 USPQ2d 1081, 1089 (Fed. Cir. 2004). The correct principles for construction are provided at MPEP § 2173.02. Definiteness of claim language must be analyzed, not in a vacuum, but in light of:

(A)The content of the particular application disclosure;

(B)The teachings of the prior art; and

(C)The claim interpretation that would be given by one possessing the ordinary level of skill in the pertinent art at the time the invention was made.

Moreover, a claim term that is not used or defined in the specification is not indefinite if the meaning of the claim term is discernible. *Bancorp Services, L.L.C. v. Hartford Life Ins. Co.*, 359 F.3d 1367, 1372, 69 USPQ2d 1996, 1999-2000 (Fed. Cir. 2004).

(VII.D) Rejection of Claims 9-19 for Nom-Statutory Double Patenting

Claims 9-19 stand rejected for non-statutory double patenting as being unpatentable over claims 1-8 of copending Application No. 11/853,017. Applcant filed a terminal disclaimer in the present application. In view of the terminal disclaimer filed in this application, Applicant contends that this ground of rejection has been obviated.

(VII.E) Rejection of claims 9 under 35 USC 101

Claims 9-19 stand rejected under 35 USC 101 as claiming non-statutory subject matter. The office action appears to argue that the apparatus of claims 9-11 and the

program storage device of claims 12-19 merely programize a formula. The office action then argues that the claims fail to meet the tangible, concrete and practical results.

The Examiner appears to be applying the State Street test for a statutory process. Under State Street, a process is patentable if it produces "a useful, tangible and concrete result". Applicants respectfully point out that the rejected claims are for an apparatus and program storage device, which are articles of manufacture not processes. Moreover, under Bilski, the State Street test is rejected and determination of whether or not processes are drawn to patentable subject matter is made by the machine-or-transformation test. The machine-or-transformation test is a two-branched inquiry; an applicant may show that a process claim satisfies § 101 either by showing that his claim is tied to a particular machine, or by showing that his claim transforms an article.

Applicant would like to respectfully point out that claims 9-11 are directed to an apparatus comprising logic. Logic is an article of manufacture comprising a processor, a series of discrete components, an integrated circuit, or the like. Applicant respectfully contends that the Office Action analysis is not appropriate for such apparatus. An apparatus comprising logic is a physical article of manufacture falling into the patentable category of manufactures under § 101 and should not be analyzed as a process. Moreover, as clearly indicated in the specification, this logic may be a processor such as a microprocessor, and may include internal memory {see para. 018}. As such, claims 9-11 are directed to an apparatus comprising logic performing stated functions on a software component.

Even if claims 9-11 were analyzed as a process, they meet the machine-or-transformation test of Bilski. First, the claims are tied to a particular machine, namely an apparatus comprising logic. Secondly, the claims claim a transformation, namely the creation of a metric for software complexity which is representative of characteristics of a software component and is useful in among other purposes: resource allocation, planning, scheduling, and cost estimation. As will be readily understood by those of ordinary skill in the art, this metric can be provided in various useful tangible forms, including: a digital

memory with the metric encoded thereon, a display or paper media with the metric visible thereon, a digital media with the metric encoded thereon.

Applicant would also like to respectfully point out that claims 12-19 are directed to a program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component. Applicant respectfully contends that the Office Action analysis is not appropriate for a program storage device which is a product of manufacture and not a process.

Even if the claims in question were treated as methods as the Examiner suggests, Applicant respectfully contends that they meet the statutory requirements of section 101. In *Schrader* the Federal Circuit determined that section 101 required a process claim to have a transformation or reduction of subject matter and that data or signals may constitute subject matter. Claim 9 includes several transformations of subject matter. First, a plurality of versions of a software component are created. As specifically provided in the specification a normalized version of the software component may be created by, for example, removing comments, converting sequences of spaces into a single space, and sorting the remaining lines in lexicographic order. Alternatively, a normalized version of the software component may be created by reformatting the program text according to a stylistic standard. Similarly, a normalized unique version may be created by eliminating duplicate lines. Creating new versions of a software component clearly comprises a transformation of subject matter – the subject matter being the program text of the software component. The newly created version is something different than the original program text of the software component. Additionally, the step of compressing the versions of the software component is also a transformation, as the compressed versions are something different than the versions created in the previous step. Then, lengths of the compressed versions are determined, again producing something new. A length of a compressed version of a software component is different from the compressed version itself. Providing a software complexity metric comprising a comparison of the lengths of the compressed versions is

yet another transformation. The software complexity metric is not a mere abstract idea, but a useful tool in the field of software development and testing for assessing the effort required for various operations pertaining to the software. Moreover, Applicant would like to respectfully point out that the metric is not merely a number, but a measure of a tangible thing, namely the complexity of a specific software component. Finally, presenting the complexity metric is another transformation, as the metric is being transformed from a signal in the logic to something tangible, concrete and useful – a presentation of the metric, such as a display or a transformation of a memory state.

With regard to the analysis of the Examiner, Applicant respectfully disagrees that Claims 9-19 are simply a formula or pure mathematics. In Bilski the Federal Circuit determined that section 101 required a process claim to be tied to a specific machine or to provide a transformation of subject matter. In Schrader, the Federal Circuit acknowledged that data or signals may constitute subject matter. In *Warmerdam* the Federal Circuit acknowledged that "if a claim requires more than the manipulation of ideas so that the process described in the claim produces something quite different, then the process might indeed describe statutory subject matter. Claims 9-19 provide transformations and produce a metric which is a useful measure of the complexity of a software component and is quite different than the software component. A software component is transformed to another version, such as a normalized version or a normalized, unique version. The versions of the software component are transformed again by compression. The compressed software component is different from the uncompressed version.

The metrics are not determined by simply calculating a ratio. The claims must be taken as a whole. The metric is created by determining at least two versions of a software component, compressing the different versions, measuring the bit length of the compressed versions, and dividing the bit length of one compressed version by the bit length of the other compressed version to accurately convey an aspect of the complexity of the original software component. Accordingly, Applicants respectfully submit that claims 9-19 are directed to statutory subject matter because they are drawn to a statutory

category - articles of manufacture. Moreover, even if they were treated as processes, they meet both branches of the Bilski test.

Accordingly, the Examiner has failed to make a prima facia case of non-statutory subject matter under 35 USC 101.

(VII.F) Rejection of Claims 9-19 under 35 USC 112, second paragraph

Claims 9-19 stand rejected under 35 USC 112, second paragraph as being indefinite.

As best understood, the Examiner appears to argue that it is not clear whether the length of the compressed versions of the software are the number of lines or the bit length.

As described in paragraph VII.C, above, "The requirement to 'distinctly' claim means that the claim must have a meaning discernible to one of ordinary skill in the art when construed according to correct principles....Only when a claim remains insolubly ambiguous without a discernible meaning after all reasonable attempts at construction must a court declare it indefinite.").*Metabolite Labs., Inc. v. Lab. Corp. of Am. Holdings*, 370 F.3d 1354, 1366, 71 USPQ2d 1081, 1089 (Fed. Cir. 2004). As defined in the MPEP, definiteness of claim language must be analyzed, not in a vacuum, but in light of:

(A)The content of the particular application disclosure;

(B)The teachings of the prior art; and

(C)The claim interpretation that would be given by one possessing the ordinary level of skill in the pertinent art at the time the invention was made.

Applicant has argued that the length of a compressed software component is well known in the art to mean its bit length. Kolmogorov complexity was defined in a 1965 paper by Andrei Kolmogorov as the length of a program in bits required to reproduce the original string. Software compression is a practical application of Kolmogrov complexity. Accordingly, the application is not indefinite as it would be clear from the

teachings of the prior art, as well as, to one of ordinary skill in the art that the length of the compressed program components would be their bit lengths.

The Examiner has concluded in error that because the specification does not define whether the length of a compressed software component means its length in lines or its bit length that the term is indefinite. The Examiner has failed to take into consideration the teachings of the prior art, notably Komogorov's paper which defines complexity of a string of information as the bit length of a program required to reproduce the original string. Moreover, the Examiner has erred in failing to consider the common usage of the term length in reference to data compression, which is bit lengh.

If, alternatively, the Examiner is arguing that there is no support for the claimed features, Applicants respectfully disagree. The claimed features are supported as follows.

Claim 9 is directed to an apparatus for determining complexity of a software component, comprising:

logic (200 in Fig. 2; paragraphs {018} and {019}) for determining a plurality of versions of the software component (P0, P1, P2; paragraphs {013} e.g. removing comments, reformatting according to a stylistic standard - both are known utilities) and for finding lengths of compressed versions of the plurality of versions of the software (C0, C1, C2; paragraph {016} as explained above this is the bit length of each component or the total number of bits, which may be determined using known methods) (this may be realized as a formatting or editing utility operating on a processor);

means for compressing each of the versions, to provide the compressed versions (210 in Fig. 2; paragraphs {015}, {018}, {019}; e.g., bzip2 compression program operating on a processor)

means for comparing the lengths of the compressed versions (220 in Fig 2; paragraphs {018}, {019}; e.g., a division utility operating on a processor); and

means for providing a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraph {010}; e.g., stored to memory, printed, returned to another program, and the like).

The logic, the compressor, and the divider all may be implemented using a stored-program-control processor, such as a microprocessor as described in paragraph {018}.

Accordingly, the Examiner has failed to make a prima facia case of indefiniteness under 35 USC 112, second paragraph.

Sincerely,

Steven E. Bach
Attorney for Applicants
Reg. No. 46,530

(VIII) Claims Appendix

Listing of Claims:

1. (Canceled)

2. (Canceled)

3. (Canceled)

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (previously presented) Apparatus for determining complexity of a software component, comprising:

logic for determining a plurality of versions of the software component and for finding lengths of compressed versions of the plurality of versions of the software;

means for compressing each of the versions, to provide the compressed versions;

means for comparing the lengths of the compressed versions; and

means for providing a software complexity metric comprising a comparison of the lengths of the compressed versions.

10. (previously presented) Apparatus for determining complexity of a software component, comprising:

logic for creating raw program text and normalized program text of the software component and for finding lengths of compressed raw program text and compressed normalized program text;

means for compressing the raw program text and the normalized program text to provide the compressed raw program text and the compressed normalized program text, respectively; and

means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text; and

means for providing a complexity metric comprising the ratio.

11. (previously presented) Apparatus for determining complexity of a software component, comprising:

logic for creating normalized program text and normalized unique program text of the software component and for finding lengths of compressed normalized program text and compressed normalized unique program text;

means for compressing the normalized program text and the normalized unique program text to provide the compressed normalized program text and the compressed normalized unique program text, respectively; and

means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text; and

means for providing a complexity metric comprising the ratio.

12. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating a plurality of versions of the software component;

compressing each of the versions, to provide compressed versions;

finding lengths of the compressed versions;

comparing the lengths of the compressed versions; and

providing a software complexity metric comprising a comparison of the lengths of the compressed versions.

13. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises raw program text.

14. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises normalized program text.

15. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises normalized unique program text.

16. (previously presented) The program storage device of claim 12, wherein the step of comparing comprises a step of finding a ratio using the length of the compressed version of raw program text and the length of the compressed version of normalized program text.

17. (previously presented) The program storage device of claim 12, wherein the step of comparing comprises a step of finding a ratio using the length of the compressed version

of normalized program text and the length of the compressed version of normalized unique program text.

18. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating raw program text and normalized program text of the software component;

compressing the raw program text and the normalized program text to provide compressed raw program text and compressed normalized program text, respectively;

finding the length of the compressed raw program text and the length of the compressed normalized program text;

finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text; and

providing a software complexity metric comprising the ratio.

19. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating normalized program text and normalized unique program text of the software component;

compressing the normalized program text and the normalized unique program text to provide compressed normalized program text and compressed normalized unique program text, respectively;

finding the length of the compressed normalized program text and the length of the compressed normalized unique program text;

finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text; and

providing a software complexity metric comprising the ratio.

(IX). Evidence appendix

There is no extrinsic evidence provided.

(X). Related proceedings appendix

There are no related proceedings.